Names: Adam Herd - 17364786, Jaime Kirk - 17922982
Project Name: Sentiment Analysis Online Chatbot
Document: Testing Documentation
Date: 07/05/2021

User Testing

We carried out user testing towards the end of this project. The server we were using to host our chatbot for this testing caused a few issues such as images not displaying and commands that didn't respond on first attempt which were not there when we tested locally so made testing less accurate as testers rated more negatively due to these issues.

Overall, user testing was a great help as it brought some big issues to our attention. Firstly, the sentiment analysis did not work correctly. It returned the same sentiment for each product no matter the reviews or overall ratings. This took quite some time to fix, as there were a number of issues causing this.

> The sentiment analysis was always 98.64% regardless of the product. Even for one with an average rating of 2.5.

Firstly, the model in our sentiment file was not returning the correct sentiments. We were not sure why this was as it had worked at our last testing. We consulted some previous tutorials we had used to get it up and running before, but nothing seemed to help. Eventually, we discovered that the issue was the fact that we were stemming the reviews that came in which gave incorrect sentiments. We stopped stemming and re-trained the model to discover that the overall accuracy had increased too. This now meant that the sentiments were returning correctly, but there was more to fix.

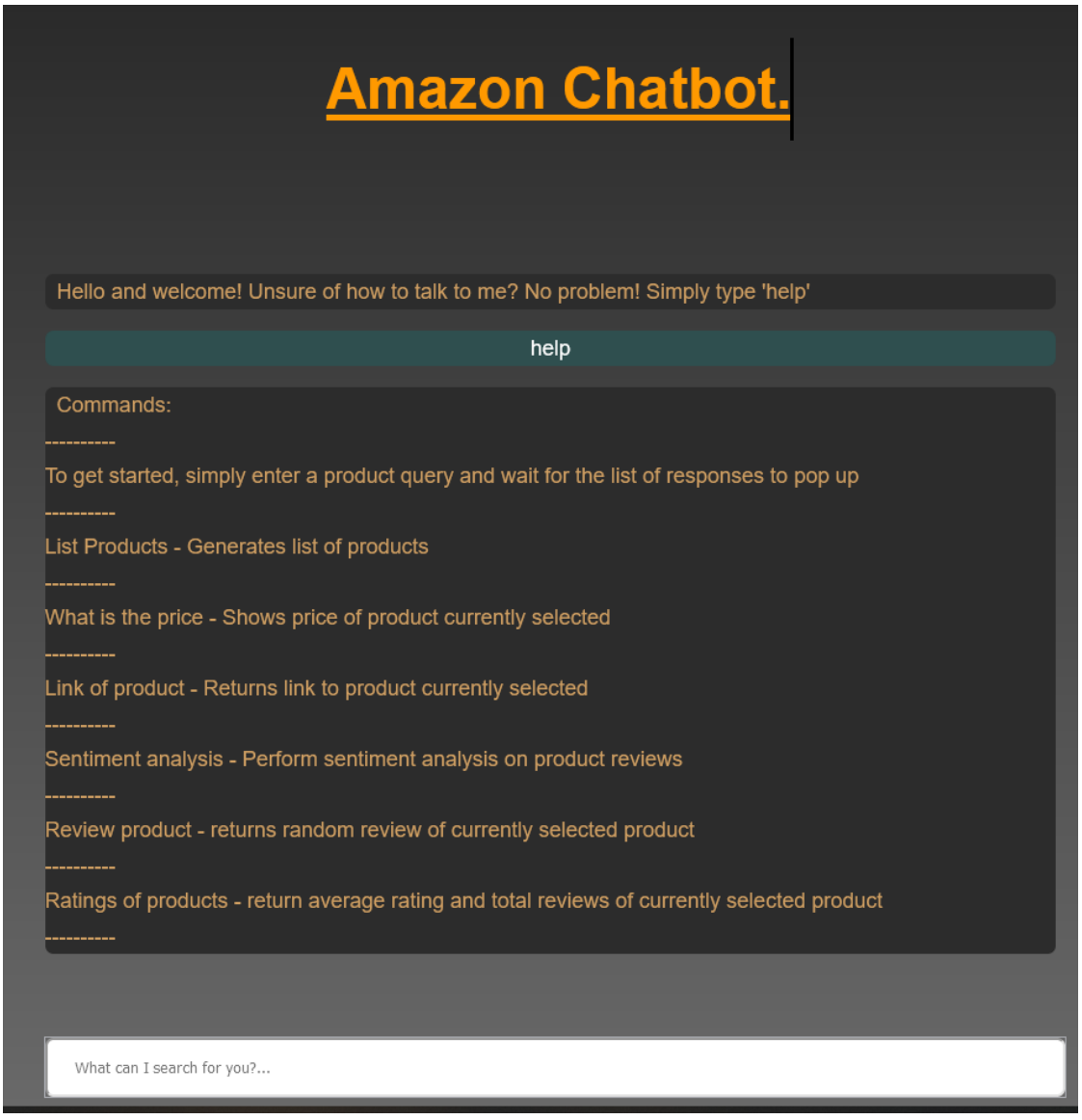
For products with more than a handful of reviews, the way we were storing them was not working. Before, we would put them all in one string and pass that string to our sentiment function. However, as the number of reviews grew, the string became too big to process. We discovered a way around this. Instead of storing all the reviews and then performing sentiment analysis on all of them at the end, we would perform sentiment analysis on each review as they came in and store the result in a list. This meant there were no overly large strings needed. Then, once all reviews were processed, we got the average sentiment from all of them and displayed that to the user.

Another useful bit of feedback we received was in relation to the help command. One user pointed out how it would be nicer to have shorter commands so they did not

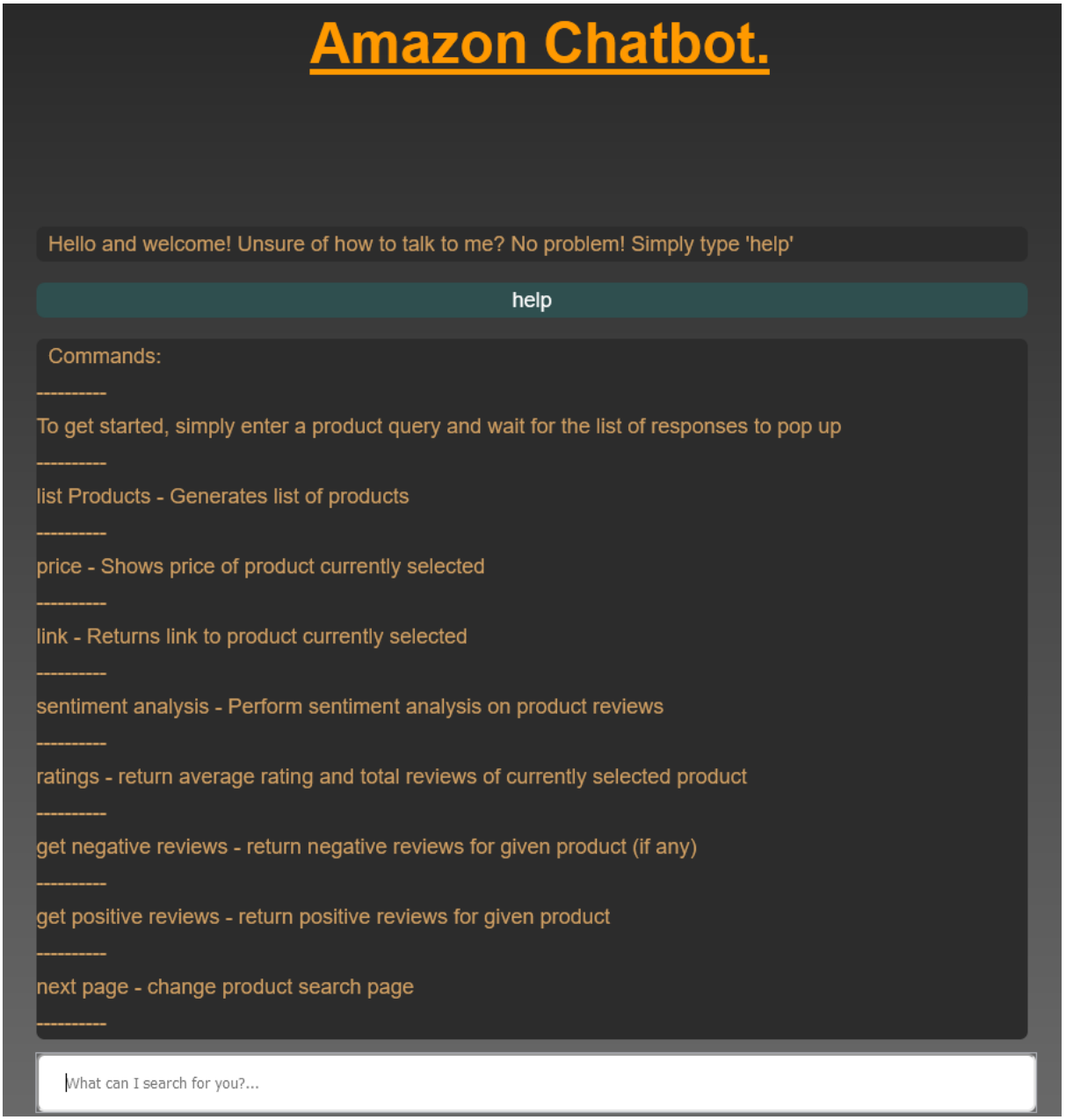
have to type multiple words for a single command. We had set in place single word commands already but had not communicated this to the user in the help command, instead we showed the multiple word versions of those commands. So we changed some of them to now show the single word versions. This did not suit all commands, but some of them.

> Commands should just be 1 word instead of sentences. Like 'what is the price' should probably just be 'price' but then explain the command better in the help section.

Now, here is what the help command looked like before. Note the multiple word commands.



And now here is what the help command looks like in its current state after the user's feedback:



Ad-hoc testing

We used a form of Ad-hoc testing to test the chatbot after implementation. This form of Ad-hoc was Buddy Testing where we both worked together to identify errors within the chatbot. We took certain areas of the code and tested different scenarios to ensure everything functioned as expected.

Some examples of bugs we found during the testing are as follows:

During testing we realised that the chatbot was not responding with its name when the user asks for the bot's name then the bot was not responding at all. This was just a simple fix and was simply just us leaving out the check for whether a command in the name tag was entered so we just extended the current check for a greeting to include a check for whether it was a name either.

```
for tg in data['replies']:
    try:
        options = tg['user_messages']
        if tg['tag'] == tag and tag == "greeting" or tg['tag'] == tag and tag == "name":
            replies = tg['responses']
            reply = random.choice(replies)
            break
```

```
{"tag": "name",
 "user_messages": ["what is your name", "what should I call you", "whats your name?", "do you have a name?", "name"],
 "responses": ["You can call me Walle.", "I'm Walle!", "People call me Walle, though I am not sure why"],
 "property_name": ""
```

Another issue that was found during this testing is that the sentiment analysis was looping and being performed multiple times however this was not spotted initially as the correct result was still being displayed and it just looked like the sentiment analysis was taking a while to calculate which we were expecting. This was only noticed when we added a print statement in the sentiment_analysis function and noticed that the function was being called multiple times. This again was a pretty straightforward fix of just taking the sentiment_analysis function call out of loop and luckily this had no effects on the rest of the chatbot and worked as expected.

```
        i = 0
        while i <= len(total_reviews) - 1:
            reviewInfo[total_reviews[i].get("body")] = total_reviews[i]
            productReviews[i] = total_reviews[i].get("body")
            productReviews[i] = productReviews[i].replace('"', "'")
            displayResults = productReviews[i]
            total_sentiment.append(sentiment_analysis(displayResults))
            i = i + 1

        j = str(int(j) + 1)

except Exception as t:
    print(repr(t))


#plotPieChart(sum(total_sentiment)/len(total_sentiment))
return str(sum(total_sentiment)/len(total_sentiment))
```

Apart from these bugs, no other bugs were found apart from some formatting issues with how text was being displayed which was easily fixed by adjusting the CSS file.

Unit Testing

Unfortunately, we ran out of time before we could adequately apply unit tests to our system. In our repository, we have set up a way of running unit tests using the library pytest. This will allow us to run any and all unit tests in the testing folder, contained within the src folder of our repository. However, setting this up to link with gitlab via CI/CD caused problems as we could not get it to recognise the pytest library. This is the cause for all of the red X's next to a lot of our commits. We left the attempted code commented out in our.yml file.

We struggled to find much online about how to properly unit test a Python flask server such as ours. Much of what we test is done through the user inputting commands on the webpage and we could not find a way to do this through unit testing. This is unfortunate as it means a large amount of potential testing could not be carried out.